

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**NETWORK ARCHITECTURE FOR SECURE
COMMUNICATIONS BETWEEN TWO CONSOLE-BASED
GAMING SYSTEMS**

Inventor(s):

Dinarte Morais
Tony Chen
Mark VanAntwerp
Boyd Multerer

ATTORNEY'S DOCKET NO. MS1-890US

TECHNICAL FIELD

This invention relates to console-based gaming systems, and more particularly, to methods for establishing secure communications between two gaming systems connected via a local area network.

BACKGROUND

Traditionally, gaming systems with a dedicated console were standalone machines that accommodated a limited number of players (e.g., 2-4 players). Personal computer-based gaming grew in popularity in part due to the ability to play games online with many remote players over the Internet. Thus, one trend for dedicated gaming systems is to provide capabilities to facilitate gaming over a network, such Internet-based online gaming and LAN-based gaming where multiple consoles are connected through a local area network (LAN).

One challenge in network gaming is to protect network traffic between any two game consoles from tampering or observation by other devices on the network. Gamers are notorious for developing creative cheating mechanisms. For example, gamers have used computers to display portions of a game map which would otherwise not be visible, or modified unprotected network traffic to give themselves advantages during play, such as perfect aim, faster players, and so on. Unfortunately, previous console-based gaming systems did not provide for secure communications.

Accordingly, there is a need for a system architecture that supports secure communications between two or more gaming systems over a local area network.

SUMMARY

A network architecture for console based gaming systems is described. The architecture allows multiple game consoles to establish secure communication over a local area network.

In the described implementation, the system architecture supports a three-phase secure communication protocol. The first phase involves generating shared keys that are unique to an authentic game console running an authentic game title. The same game consoles running the same games will create the same shared keys. In the second phase, a "client" console attempts to discover existing game sessions being hosted by a "host" game console by broadcasting a request over the local area network. The broadcast request expresses a desire to join in playing the game at the next convenient opportunity. The broadcast request is protected using the shared keys. If the host console agrees to let the client console play, the host console generates session keys that are returned securely to the client console. The third phase involves a key exchange in which the client and host consoles exchange data used to derive one or more secrets. The key exchange is protected using the session keys. The secrets are then used to establish a secure point-to-point link between the two consoles for ongoing communication.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates a gaming system with a game console and one or more controllers.

Fig. 2 is a block diagram of the gaming system.

Fig. 3 illustrates a network gaming system in which the Fig. 1 gaming system is connected via a local area network to other gaming systems.

1 Fig. 4 illustrates a first phase of a secure communication process in which a
2 game console running an authentic game title generates shared secret keys.

3 Fig. 5 illustrates a second phase of the secure communication process in
4 which a client game console discovers a host game console on the local area
5 network.

6 Fig. 6 illustrates a third phase of the secure communication process in
7 which the client and host game consoles establish a point-to-point communication
8 link.

9 10 **DETAILED DESCRIPTION**

11 The following discussion is directed to console-based gaming systems and
12 techniques for establishing secure communications between multiple gaming
13 systems connected via a local area network (LAN). The discussion assumes that
14 the reader is familiar with basic cryptography principles, such as encryption,
15 decryption, authentication, hashing, and digital signatures. For a basic
16 introduction to cryptography, the reader is directed to a text written by Bruce
17 Schneier and entitled, "Applied Cryptography: Protocols, Algorithms, and Source
18 Code in C," published by John Wiley & Sons, copyright 1994 (second edition
19 1996), which is hereby incorporated by reference.

20 21 **Gaming System**

22 Fig. 1 shows an exemplary gaming system 100. It includes a game console
23 102 and up to four controllers, as represented by controllers 104(1) and 104(2).
24 The game console 102 is equipped with an internal hard disk drive and a portable
25 media drive 106. The portable media drive 106 supports various forms of portable

1 storage media as represented by optical storage disc 108. Examples of suitable
2 portable storage media include DVD, CD-ROM, game discs, game cartridges, and
3 so forth.

4 The game console 102 has four slots 110 on its front face to support up to
5 four controllers, although the number and arrangement of slots may be modified.
6 A power button 112 and an eject button 114 are also positioned on the front face
7 of the game console 102. The power button 112 switches power to the game
8 console and the eject button 114 alternately opens and closes a tray of the portable
9 media drive 106 to allow insertion and extraction of the storage disc 108.

10 The game console 102 connects to a television or other display (not shown)
11 via A/V interfacing cables 120. A power cable 122 provides power to the game
12 console. The game console 102 may further be equipped with internal or
13 externally added network capabilities, as represented by the cable or modem
14 connector 124 to facilitate access to a network, such as a local area network
15 (LAN) or the Internet.

16 Each controller 104 is coupled to the game console 102 via a wire or
17 wireless interface. In the illustrated implementation, the controllers are USB
18 (Universal Serial Bus) compatible and are connected to the console 102 via serial
19 cables 130. The controller 102 may be equipped with any of a wide variety of
20 user interaction mechanisms. As illustrated in Fig. 1, each controller 104 is
21 equipped with two thumbsticks 132(1) and 132(2), a D-pad 134, buttons 136, and
22 two triggers 138. These mechanisms are merely representative, and other known
23 gaming mechanisms may be substituted for or added to those shown in Fig. 1.

24 A memory unit (MU) 140 may be inserted into the controller 104 to
25 provide additional and portable storage. Portable memory units enable users to

1 store game parameters and transport them for play on other consoles. In the
2 described implementation, each controller is configured to accommodate two
3 memory units 140, although more or less than two units may be employed in other
4 implementations.

5 The gaming system 100 is capable of playing, for example, games, music,
6 and videos. With the different storage offerings, titles can be played from the hard
7 disk drive or the portable medium 108 in drive 106, from an online source, or from
8 a memory unit 140. A sample of what the gaming system 100 is capable of
9 playing back includes:

- 10
11 1. Game titles played from CD and DVD discs, from the hard disk drive,
12 or from an online source.
- 13 2. Digital music played from a CD in the portable media drive 106, from a
14 compressed file on the hard disk drive (e.g., Windows Media Audio
15 (WMA) format), or from online streaming sources.
- 16 3. Digital audio/video played from a DVD disc in the portable media drive
17 106, from a file on the hard disk drive (e.g., Windows Media Video
18 (WMV) format), or from online streaming sources.
- 19

20 Fig. 2 shows functional components of the gaming system 100 in more
21 detail. The game console 102 has a central processing unit (CPU) 200 and a
22 memory controller 202 that facilitates processor access to various types of
23 memory, including a flash ROM (Read Only Memory) 204, a RAM (Random
24 Access Memory) 206, a hard disk drive 208, and the portable media drive 106.
25 The CPU 200 is equipped with a level 1 cache 210 and a level 2 cache 212 to

1 temporarily store data and hence reduce the number of memory access cycles,
2 thereby improving processing speed and throughput.

3 The CPU 200, memory controller 202, and various memory devices are
4 interconnected via one or more buses, including serial and parallel buses, a
5 memory bus, a peripheral bus, and a processor or local bus using any of a variety
6 of bus architectures. By way of example, such architectures can include an
7 Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA)
8 bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association
9 (VESA) local bus, and a Peripheral Component Interconnect (PCI) bus.

10 As one suitable implementation, the CPU 200, memory controller 202,
11 ROM 204, and RAM 206 are integrated onto a common module 214. In this
12 implementation, ROM 204 is configured as a flash ROM that is connected to the
13 memory controller 202 via a PCI (Peripheral Component Interconnect) bus and a
14 ROM bus (neither of which are shown). RAM 206 is configured as multiple DDR
15 SDRAM (Double Data Rate Synchronous Dynamic RAM) modules that are
16 independently controlled by the memory controller 202 via separate buses (not
17 shown). The hard disk drive 208 and portable media drive 106 are connected to
18 the memory controller via the PCI bus and an ATA (AT Attachment) bus 216.

19 A 3D graphics processing unit 220 and a video encoder 222 form a video
20 processing pipeline for high speed and high resolution graphics processing. Data
21 is carried from the graphics processing unit 220 to the video encoder 222 via a
22 digital video bus (not shown). An audio processing unit 224 and an audio codec
23 (coder/decoder) 226 form a corresponding audio processing pipeline with high
24 fidelity and stereo processing. Audio data is carried between the audio processing
25 unit 224 and the audio codec 226 via a communication link (not shown). The

1 video and audio processing pipelines output data to an A/V (audio/video) port 228
2 for transmission to the television or other display. In the illustrated
3 implementation, the video and audio processing components 220-228 are mounted
4 on the module 214.

5 Also implemented on the module 214 are a USB host controller 230 and a
6 network interface 232. The USB host controller 230 is coupled to the CPU 200
7 and the memory controller 202 via a bus (e.g., PCI bus) and serves as host for the
8 peripheral controllers 104(1)-104(4). The network interface 232 provides access
9 to a network (e.g., LAN, Internet, etc.) and may be any of a wide variety of
10 various wired or wireless interface components including an Ethernet card, a
11 modem, a Bluetooth module, a cable modem, and the like.

12 The game console 102 has two dual controller support subassemblies
13 240(1) and 240(2), with each subassembly supporting two game controllers
14 104(1)-104(4). A front panel I/O subassembly 242 supports the functionality of
15 the power button 112 and the eject button 114, as well as any LEDs (light emitting
16 diodes) or other indicators exposed on the outer surface of the game console. The
17 subassemblies 240(1), 240(2), and 242 are coupled to the module 214 via one or
18 more cable assemblies 244.

19 Eight memory units 140(1)-140(8) are illustrated as being connectable to
20 the four controllers 104(1)-104(4), i.e., two memory units for each controller.
21 Each memory unit 140 offers additional storage on which games, game
22 parameters, and other data may be stored. When inserted into a controller, the
23 memory unit 140 can be accessed by the memory controller 202.
24
25

1 A system power supply module 250 provides power to the components of
2 the gaming system 100. A fan 252 cools the circuitry within the game console
3 102.

4 A console user interface (UI) application 260 is stored on the hard disk
5 drive 208. When the game console is powered on, various portions of the console
6 application 260 are loaded into RAM 206 and/or caches 210, 212 and executed on
7 the CPU 200. The console application 260 presents a graphical user interface that
8 provides a consistent user experience when navigating to different media types
9 available on the game console.

10 The game console 102 implements a cryptography engine to perform
11 common cryptographic functions, such as encryption, decryption, authentication,
12 digital signing, hashing, and the like. The cryptography engine may be
13 implemented as part of the CPU 200, or in software stored in memory (e.g., ROM
14 204, hard disk drive 208) that executes on the CPU, so that the CPU is configured
15 to perform the cryptographic functions.

16 The gaming system 100 may be operated as a standalone system by simply
17 connecting the system to a television or other display. In this standalone mode,
18 the gaming system 100 allows one or more players to play games, watch movies,
19 or listen to music. However, with the integration of network connectivity made
20 available through the network interface 232, the gaming system 100 may further
21 be operated as a participant in a larger network gaming community. This network
22 gaming environment is described next.

Network Gaming

Fig. 3 shows an exemplary network gaming environment 300 in which multiple gaming systems 100(1), 100(2),..., 100(g) are interconnected via a local area network 302. LAN 302 may be implemented using any one or more of a wide variety of conventional communications media including both wired and wireless media. As one example, the LAN 302 represents an Ethernet cross-over cable or an Ethernet hub. Other possible implementations of LAN 302 include FireWire (specifically, IEEE 1394) and universal serial bus (USB).

Many game titles are scalable to support multiple players on different consoles. Interconnecting multiple gaming systems via the LAN 302 enables multiple players to compete in the same game. For example, some game titles allow up to 16 different players, which can be composed of any combination of players and gaming systems (e.g., from one player each on 16 gaming systems to four players each on four gaming systems).

A network-based game is hosted by one of the game consoles. For discussion purposes, suppose that gaming system 100(1) hosts the game for client gaming systems 100(2), ..., 100(g). In this context, the hosting function is one of coordinating player access to the multiplayer game. The "hosting" metaphor is not meant to imply that one game console serves the game content to other game consoles. Each game console 100 has its own copy of the game stored either on the hard disk drive 208 (e.g., purchased and downloaded an authentic game title from a vendor's website) or the portable storage medium as represented by a digital video disk (DVD) 108(1), 108(2), ..., 108(g).

Secure Communication Protocol

All game consoles involved in a network gaming situation initially establish secure communication links between one another. Secure communications means that data can be transferred between two entities in a manner that is resistant to tampering. The transferred data can be authenticated by the entities to ensure that a trusted entity did indeed send that particular data. Secure communications may or may not involve encryption, which effectively renders the transmitted data blind to an observer. If encryption is used, it may occur before or after authentication.

With reference to Fig. 3, each client game console 100(2)-(g) that wants to participate seeks permission from the host gaming system 100(1). If the host is willing, secure communications are established between the requesting client console and the host console. Establishment of a secure communications link is accomplished by a three phase process. Briefly, the first phase involves generation of shared secret keys that only an authentic game console running an authentic game title will know. The second phase concerns session discovery in which a client console attempts to discover existing game sessions being hosted by a host game console. During session discovery, the client console broadcasts a discovery request over the local area network. The discovery request expresses a desire to join the LAN-based game at the next convenient opportunity. The discovery request is protected using the shared keys. If the host console agrees to let the client console play, the host console generates a discovery reply containing session keys. The discovery reply is returned to the client console via broadcast, directly, or indirectly. The third phase involves a key exchange in which the client and host consoles exchange data used to derive one or more secrets. The key exchange is protected using the session keys. The secrets are then used to

1 establish a secure point-to-point link between the multiple consoles for ongoing
2 communication.

3 These phases will be described below in more detail according to one
4 possible implementation of the secure communication protocol.

6 Phase I: Generate Shared Secret Keys

7 Fig. 4 shows the first phase 400 of the secure communication protocol in
8 which a shared secret key that is unique to an authentic gaming system running an
9 authentic game title is generated. The console stores a secret key 402 in a secure
10 location, such as in a protected portion of ROM 204. Additionally, each game title
11 certified to play on the gaming system has an associated unique, randomly-
12 generated title key 404. The title key 404 is the same for every copy of the same
13 game title. Different titles have different title keys. The title key 404 may be
14 stored as part of the game title, such as on the DVD or with a softcopy of the game
15 title that was legitimately purchased online and downloaded for storage on the
16 hard disk drive of the game console.

17 A LAN key generator 406 uses the console-based key 402 and the title-
18 based key 404 to derive a LAN key 408. In one implementation, the LAN key
19 generator 406 computes a one-way cryptographic function using the two keys 402
20 and 404 during a network stack initialization. As one example of a suitable
21 cryptographic function, the LAN key generator 406 concatenates the keys 402 and
22 404 and performs an HMAC-SHA-1 (Hashed Message Authentication Code—
23 Secure Hash Algorithm 1) operation on them to produce the LAN key 408.

24 The LAN key 408 is then used to produce shared secret keys. More
25 particularly, when the network stack is initialized, a broadcast key generator 410

1 uses the LAN key 408 to generate a title broadcast encryption key 412 and a title
2 broadcast signature key 414. As one example, the generator 410 may create keys
3 for symmetric ciphers, asymmetric ciphers, hashing algorithms, and digest
4 algorithms.

5 Generation of the broadcast keys 412 and 414 is performed once during
6 initialization. All gaming systems running the same title will generate the same
7 pair of keys. Any gaming system running a different title on the same LAN will
8 have a different pair of keys. The combination of keys prevents anyone from
9 discovering how to intercept or modify broadcast messages for a particular game
10 simply by being able to read the game title.

11 Phase II: Session Discovery

12 Fig. 5 shows the second phase 500 of the secure communication protocol in
13 which a client console attempts to discover whether any other consoles on the
14 LAN are currently hosting a multiplayer game session of a particular title. With
15 only a few consoles connected on a small local area network, there is not expected
16 to be more than one or two consoles currently hosting a particular game. With
17 small numbers of consoles, there is no third-party matchmaking service and hence
18 the client console is left to discover suitable hosts on its own through a broadcast
19 technique. It is noted, however, that in the future the number of available game
20 sessions might be quite large and a dedicated matchmaking service may be used to
21 find a suitable match.
22

23 The second phase is described in the context of a general flow of operations
24 performed by a requesting client console 100(2) and a potential host console
25

1 100(1). The operations are illustrated beneath the client and host consoles to
2 convey where such operations are performed.

3 At operation 502, the client console 100(2) generates a request REQ
4 containing a request message and an optional nonce N_i (the initiator's nonce). At
5 operation 504, the client console encrypts the request using a symmetric key
6 cipher E (e.g., DES) that employs the title broadcast encryption key 412 (denoted
7 as K_E), as follows:

$$8 \quad REQ^E = E_{K_E}[\text{request message}, N_i].$$

9
10
11 At operation 506, the client console digitally signs the request using the
12 title broadcast signature key 414 (denoted as K_S), as follows:

$$13 \quad REQ^{ES} = S_{K_S}[REQ^E].$$

14
15
16 At operation 508, the client console broadcasts the encrypted and signed
17 request REQ^{ES} to other consoles on the LAN 302. The host console 100(1) has
18 previously opened a UDP (User Datagram Protocol) socket and is listening for
19 discovery broadcast messages over the LAN. Consoles running the same game
20 title will recognize the request because they are able to generate the same set of
21 title broadcast keys 412 and 414, and hence can authenticate and decrypt the
22 message. Consoles running different game titles, however, will disregard the
23 message. Assume that one host console exists and is running the same game title
24 (operation 510). Accordingly, the host console is likewise able to derive the title
25

1 broadcast encryption key 412 and the title broadcast signature key 414 as provided
2 in phase I (Fig. 4).

3 At operation 512, the host console authenticates the request using the same
4 title broadcast signature key 414. At operation 514, the host console decrypts the
5 message using the same title broadcast encryption key 412, as follows:

$$7 \quad \text{REQ} = D_{KE}[\text{REQ}^E].$$

8
9 At operation 516, the host console decides whether to allow the requesting
10 client console to participate in the network game. Factors affecting this decision
11 include the number of current players, number of players supported by the game,
12 current status of the game (i.e., just beginning, middle of session, etc.), and so on.
13 Assuming that the host console will accept a new client console with one or more
14 players, the host console generates a unique pair of keys used to distinguish
15 between different sessions (operation 518). These keys include a network key
16 identifier NKID and a network key exchange key NKEY. The identifier NKID is
17 simply a name for the key NKEY. In one implementation, the session keys are
18 created by a random number generator. These keys are used in the key exchange
19 phase described below with reference to Fig. 6. The session key NKEY will be
20 used in the future to authenticate the contents of key exchange packets exchanged
21 between the two consoles, whereas the key identifier NKID is used to specify the
22 game session to which the source console is connecting. It is noted that the
23 session keys NKID and NKEY may be pre-generated prior to receiving any
24 request from a client console.
25

1 At operation 520, the host console creates a reply to the discovery request.
2 The reply contains, at a minimum, a reply message, the NKID / NKEY pair, and a
3 network address NADDR of the host game console. The reply message may
4 include the initiator's nonce Ni, a responder's nonce Nr, and game specific data.
5 At operation 522, the host console encrypts the reply using the title broadcast
6 encryption key 412, as follows:

$$8 \quad \text{REPLY}^E = E_{KE}[\text{reply message, NKID, NKEY, NADDR}].$$

9
10 At operation 524, the client console signs the reply using the title broadcast
11 signature key 414 (denoted as K_S), as follows:

$$12 \quad \text{REPLY}^{ES} = S_{K_S}[\text{REPLY}^E].$$

13
14
15 At operation 526, the host console broadcasts the reply across the local area
16 network 302 to other consoles, including the requesting client console 100(2). At
17 operation 528, the client console authenticates and decrypts the reply. Every host
18 console responds with its own encrypted broadcast reply containing the network
19 address, as well as an encryption key and identifier specific to that game session.
20 Having discovered one or more hosts available on the network, the client
21 communicates with the desired host using session key pair of that host. Any other
22 session key pairs returned by other potential hosts are discarded.
23
24
25

Phase III: Key Exchange

Fig. 6 shows the third phase 600 of the secure communication protocol in which the client and host perform a secure key exchange to establish a secure point-to-point link for ongoing communication. After game session discovery, the client and host consoles share a title broadcast encryption key 412 and a title broadcast signature key 414, as well as a session key pair NKID/NKEY for a particular game session. Now, the client and host consoles exchange data used to derive one or more keys that are used to secure point-to-point communication. In the described implementation, the consoles use Diffie-Hellman exponentiation operations to derive a new secret that is shared between the two consoles without transmitting that secret across the LAN. The key exchange is protected using the session key.

For the discussion purposes, suppose the client console is the initiator of the key exchange and the host is the responder. When the client sends the first packet, the client notices that there is no security association established between the game client and game host. The packet is placed on a packet queue and the key exchange process is started.

At operation 602, the client creates an initial key exchange packet KeyExInit. The packet includes secrets used to derive symmetric keys used to secure communication in a point-to-point communication link. As one exemplary implementation, the key exchange packet is a UDP packet with a payload containing the following data:

Data Field	Definition
NonceInit	This field contains a random nonce created by the

initiator. It is used to validate replies and compute keys.

NonceResp This field contains a random nonce created by the responder. It is used to validate replies and compute keys.

g^X This field contains the Diffie-Hellman exponentiation provided by the initiator.

NKID This field specifies the key identifier that identifies the game session key and that both consoles have by virtue of the second phase session discovery.

NADDR This field contains the complete network address information of the sender of the message.

Time This field contains a time value that is used to limit replay attacks. Every time a key exchange packet is transmitted (or retransmitted for timeouts), this value is incremented by one unit and that time is transmitted in the key exchange packet. Should the initiator or responder reboot and attempt to rejoin the same game session (because it reacquired the same NKID / NKEY pair from the game host via broadcast when it came back up), its system time will have moved forward by many thousands if not millions of units. The purpose of this scheme is to prevent the replay of key exchange initiator packets disrupting the security association between a pair of consoles.

Hash This field contains a hash digest of the key exchange packet.

At operation 604, the client console computes a hash digest of the entire message using the session key NKEY and a hashing function H, as follows:

$$\text{HashInit} = H_{\text{NKEY}} [\text{NonceInit}, g^X, \text{NKID}, \text{NADDR}, \text{Time}].$$

1
2 Accordingly, creation of the initial key exchange packet KeyExInit
3 involves generating a random nonce "NonceInit", filling in the computed Diffie-
4 Hellman value " g^X ", filling in the NKID and NADDR fields, and incrementing
5 and setting the "Time" field. All other fields are zeroed. The hash digest of all
6 preceding fields ("HashInit") is placed in the "Hash" field of the key exchange
7 packet so that the packet can be authenticated by the receiving console (e.g., host
8 console). The resulting packet is as follows:

9
10 KeyExInit: [NonceInit, g^X , NKID, NADDR, Time, HashInit].
11

12 At operation 606, the client console sends the initial key exchange packet
13 KeyExInit to the host console. When this message is received, the host console
14 uses the key identifier in the NKID field to identify the key NKEY of the
15 particular game session. The session key is then used to authenticate the message
16 as originating from the client console (operation 608). More particularly, the host
17 uses the session key NKEY to compute a hash digest of the contents and compare
18 that digest to the attached "HashInit" value received in the initial key exchange
19 packet.

20 At operation 610, the host console creates a responsive key exchange
21 packet (denoted as "KeyExResp"). In the continuing example, the host console
22 generates a random nonce "NonceResp", fills in its computed value from the
23 Diffie-Hellman function g^Y , fills in the NKID and NADDR fields, and increments
24 and sets the "Time" field. The "NonceInit" field is copied from the initial
25 message.

1 At operation 612, the host console hashes the response packet using the
2 session key NKEY, as follows:

$$3$$
$$4 \text{ HashResp} = H_{\text{NKEY}} [\text{NonceInit}, \text{NonceResp}, g^Y, \text{NKID}, \text{NADDR}, \text{Time}].$$
$$5$$

6 The hash digest is placed in the "Hash" field of the key exchange packet.
7 The response packet now has the following contents:

8

$$9 \text{ KeyExResp: } [\text{NonceInit}, \text{NonceResp}, g^Y, \text{NKID}, \text{NADDR}, \text{Time},$$
$$10 \text{ HashResp}].$$
$$11$$

12 At operation 614, the host console sends the response key exchange packet
13 KeyExResp to the client console. Upon receipt, the client console uses the key
14 identifier in the NKID field to identify the session key NKEY. The session key
15 NKEY is then used to authenticate the message as originating from the host
16 console (operation 616). That is, the client uses the session key NKEY to compute
17 a hash digest of the response packet and compare that digest to the attached
18 "HashResp" value received in the response packet.

19 At this point, both the host and the client have enough data to compute
20 security association keys (operation 618). Security association keys are used to
21 secure point-to-point communication between two consoles. In this example, each
22 console uses the two Diffie-Hellman exponentiations to compute function g^{XY} .
23 Each console can then compute four different digests using NonceInit,
24 NonceResp, and g^{XY} as input, as well as the session key NKEY. These digests are
25 used to form the security association keys.

1 Once the client has the security association keys, it is free to transmit any
2 packets that have been waiting for key exchange to complete. The host console,
3 however, is not free to do so even though it has the same set of keys because it
4 cannot be sure that its response message KeyExResp was not lost. The host waits
5 until it either receives a KeyExAck message or until it receives a packet
6 authenticated with the computed security association key (operation 620).

7 In the common case, the client console sends a packet to the host and thus,
8 the key exchange process consists of just two messages—KeyExInit and
9 KeyExResp. Should the client not have a packet to send, it will send an artificial
10 acknowledge message (denoted as “KeyExAck”). This message differs from the
11 two other key exchange messages in that the message is hashed using the
12 computed security association key instead of the session key NKEY.

13 From this point forward, the two consoles use the security association keys
14 to secure communications. All network packets that need to be transmitted to the
15 other consoles are authenticated after optionally being encrypted, with the
16 receiving console verifying the authentication data before decrypting the packet
17 contents. Once a game session is finished, either console can stop accepting key-
18 exchange requests from any console using this session key pair NKID/NKEY.
19 The duration of the session keys is expected to be anywhere from a few minutes to
20 a few hours.

21 22 Conclusion

23 Although the invention has been described in language specific to structural
24 features and/or methodological acts, it is to be understood that the invention
25 defined in the appended claims is not necessarily limited to the specific features or

1 acts described. Rather, the specific features and acts are disclosed as exemplary
2 forms of implementing the claimed invention.
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25